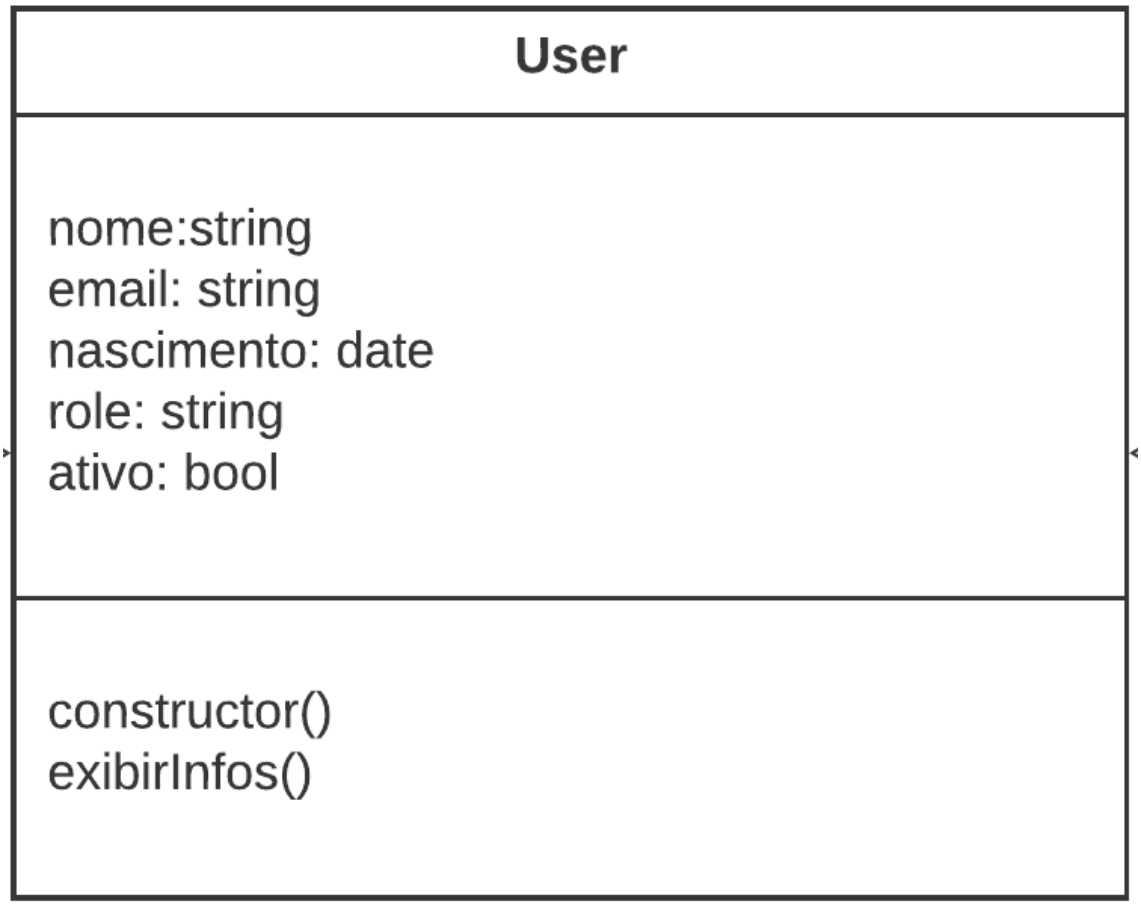
Durante a aula, vimos que um dos campos do objeto `User` no diagrama UML utilizava não o formato `String` ou `Number`, mas sim o formato `Date`.



Vamos ver um pequeno resumo do que se trata.

Como o nome sugere, o `Date` é um objeto utilizado para trabalhar as datas e o tempo em JavaScript. No dia a dia de desenvolvimento é muito comum precisarmos utilizar a informação da data e hora para realizar alguma tarefa ou, especialmente, lidar com dados. No entanto, em JavaScript, essa mesma informação pode assumir diferentes tipos.
Vamos supor que você decida utilizar a variável `tempo` para guardar uma informação da data e, para obter o valor, você decida chamar o método `Date()` do objeto Date:

```
let dataEHora = Date() // utilizando a chamada de função diretamente
console.log(dataEHora)
 // Mon Jan 31 2022 23:44:05 GMT-0300 (Brasilia Standard Time)
console.log(typeof dataEHora)
 // string
```

Como podemos ver, a saída da variável é a data completa (no padrão de Brasília, nesse cenário específico) e o tipo da variável dataEHora é `string`.
Em outro cenário, suponhamos que você decida utilizar a mesma variável `dataEHora` para guardar a informação da data mas, diferentemente do primeiro caso onde você chamou apenas o método Date() para recuperar a informação, você agora crie uma instância do objeto Date.

```
let dataEHora = new Date() // utilizando o construtor do objeto Date
console.log(dataEHora)
 // 2022-02-01T02:46:51.517Z
console.log(typeof dataEHora)
 // object
```

A informação então recuperada é a mesma mas os tipos são diferentes. Mas qual o propósito disso?

A razão por trás desse comportamento é que, quando utilizamos um construtor, também temos acesso a todos os métodos do objeto `Date`. Podemos ver melhor essa diferença nos exemplos a seguir:

```
let dataComConstrutor = new Date()
let data1 = dataComConstrutor.getDate()
console.log(data1) //31
```

Acima, utilizamos o método `getDate()`, que já existe em qualquer objeto criado a partir de `Date()`, para obter o dia do mês.

```
let dataFuncao = Date()
let data2 = dataFuncao.getDate()
 //TypeError: dataFuncao.getDate is not a function
```

Já no exemplo acima, vemos que uma data gerada através da chamada da função `Date()` diretamente não pode acessar os demais métodos do objeto `Date`, ao passo que ao gerarmos uma nova instância desse objeto é possível acessar todos os métodos definidos dentro deste objeto, como por exemplo `getDate()`.

Alguns exemplos de outros métodos disponíveis no objeto Date são:

- `.getMillisseconds()`
- `.getSeconds()`
- `.getMinutes()`
- `.getHours()`
- `.getDay()`
- `.getMonth()`
- `.getFullYear()`

Em seguida vamos estudar mais sobre o `new`, funções construtoras e métodos de objetos. Porém você pode sempre acessar a documentação do MDN e testar os vários formatos de data possíveis.